University Politehnica of Bucharest
Faculty of Automatic Control and Computers
Computer Science and Engineering Department

BACHELOR THESIS

Mobile Gateway for Wireless Sensor Networks utilizing drones

by

Ioan Deaconu

Supervisor: As. Drd. Ing. Andrei Voinescu

Bucharest, September 2014

Contents

Abstract

Keywords Wireless Sensor Networks, task, scheduling, graph cuts

Acknowledgements

Chapter 1

Introduction

Thesis Intro - no more than 3 pages.

Chapter 2

Related Work

Research has been done in the area of wireless sensor networks with mobile gateways, but very few include the same constraints as in our networks. Such constrains consits of long distances between nodes, simple, fast implementation and the posibility to run

Where most research focuses on routing packets to a mobile sink, we consider applications where network lifetime and low-maintenance are more important, and as such energy consumption on the nodes must be kept to a minimum (in order to have 3+ years of network lifetime with no maintenance/battery changes). Shifting the energy consumption to the sink is a direct consequence of this decision, under the assumption that there exists a flight path that allows the gateway to collect information from all the nodes.

2.1 Standard WSN Protocols

The protocols implemented in Wireless Sensor Network are based on sourounding node discovery in order to build a topology and find the best way they can multihop data to the gateway. This approach works best in a static environment, but in a dynamic environment or an eviroment were the distance between nodes is too big or the time between two data packets is too big, the network convergence will be slow or not even possible.

2.2 Crop Monitoring

[VSB+11]

A reasearch of using a drone for crop monitoring has been conducted at a vineyard. Their system was comprised of a unmanned quadcopter, an Arduino board with a GPRS module for long distance communication with the drone

and ZigBee and Crossbows TelosB as wireless sensing nodes. The drone was not controlled via the long-distance link, but through a Spektrum DX7SE 2.4 GHz remote control.

They demonstrated that a preprogramed UAV can be used to monitor multiple crops where a standard WSN could not be deployed because of the unique constrains imposed by the environment.

The cost of the implementation was relatively high compared to ours, the remote is 300$, the same as the entire drone that we propose and the TelosB is 99$. This data suggests that for their experiment the drone, communication module and the remote control were half the cost of the equipment.

Another problem was that they were not saving the data localy, but sending it back to the base station where it was proccesed and saved. This can represent a problem because the system cannot function propely unless a base station is supplied.

2.3 Aware platform

[OBLC+07]

The Aware platform, proposed by Ays. Egl Tysz Erman, Lodewijk Van Hoesel and Paul Havinga from University of Twente, is a platform that integrates WSNs, UAVs, and actuators into a disaster response setting and provides facilities for event detection, autonomous network repair by UAVs, and quick response by integrated operational forces.

They use multiple UAVs to deploy new nodes that will replace the damaged ones and check if they function. The entire system still relies on a sink to collect the data and to send them to a base station.[EHHW08]

Chapter 3

Hardware Platform

In this chapter we will present the hardware platforms used.

Because we wanted to emphasise not only a new way of aquiring data, but also a simple and low cost one, we have selected The Parrot AR.Drone 2.0 as the work horse that will carry the mobile gateway which communicates with the nodes from our Sparrow Family. The drone provides several key features that we require, such as a Linux embedded system, a mobile platform with the possibility of autonomous flying and sufficient flight time for our needs.

chestii ..... help .. anybody

3.1 The Parrot AR.Drone 2.0

[Par12]

Parrot AR.Drone is a WiFi radio controlled flying quadcopter built by the French company Parrot. The original drone was released in 2010 and in 2012 it was replaced by version 2.0. Since the launch of the original AR.Drone, more the half a milion units have been sold, making it one of the, if not, the most popular drone on the market.[BRA13]

The reason of its success is not entirely due to the relatively low price of around 300$ but also because it is very easy to learn how to control the drone and also because of the USB port that accommodate any device using that interface and the Linux operating system, making it incredibly versatile and very easy to integrate in our system.

Because of those reasons, the Drone has a number of after-market modules that can be attached to it, such as the Flight Recorder GPS Module. This module has a built-in storage of 4GB for video recording purposes and a built in GPS receiver. This allows the drone to follow a predetermined path of waypoints and

Figure 3.1: *The Parrot AR.Drone 2.0*

to return back from where it took off automatically, all within the limit of the Wi-Fi connection with the control device.

The arrot AR.Drone 2.0 specifications are :

- 1GHz 32 bit ARM Cortex A8 processor with 800MHz video DSP TMS320DMC64x
- Linux 2.6.32
- 1Gbit DDR2 RAM at 200MHz
- USB 2.0 high speed for extensions
- Wi-Fi b,g,n
- 3 axis gyroscope 2000/second precision
- 3 axis accelerometer +-50mg precision
- 3 axis magnetometer 6 precision
- Pressure sensor +/- 10 Pa precision
- Ultrasound sensors for ground altitude measurement
- 60 fps vertical QVGA camera for ground speed measurement
- 30 fps 720p front mounted camera

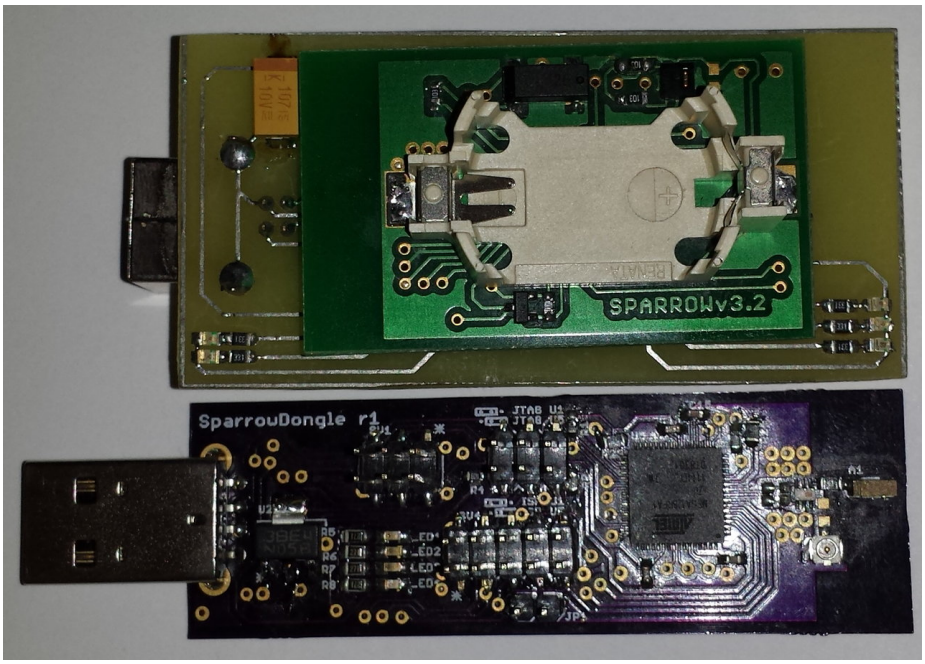

Figure 3.2: *The Sparrow Dongle next to the SparrowV3.2*

3.2 The Sparrow Family

The Sparrow Family, composed of the Sparrow Dongle and SparrowV3.2, use a 2,4 GHz wireless network as a medium of communication.

At heart, soul and brain of this family lies the ATMega128RFA1. It is an 8bit micro-controller from Atmel that has an on-chip 2.4 GHz wireless transceiver. On-chip transceivers occupy no extra space and require little extra electronics to operate, making the resulting boards very small. The on-chip transceiver allows more energy-efficient operating modes, and facilitate higher bandwidth transfers between the micro-controller's main memory and the transceiver frame-buffer, all of which are important improvements in the field of wireless sensor networks.

The signal received or sent by the wireless transceiver can be boosted by attaching an external antenna. For example, in an ideal situation, with no interferences from the outside world , an 8-dBi omni-directional antenna mounted on both communication devices would amount to an around 200 meters of communication range, well over the 70 meters measured with the default antennas.

3.2.1 The Sparrow Dongle

The link between the the wireless sensor networks and the rest of the digital world, the Sparrow Dongle is the gateway of the Sparrow Family. The Dongle can be connected to any device that has an USB port and can support USB CDC

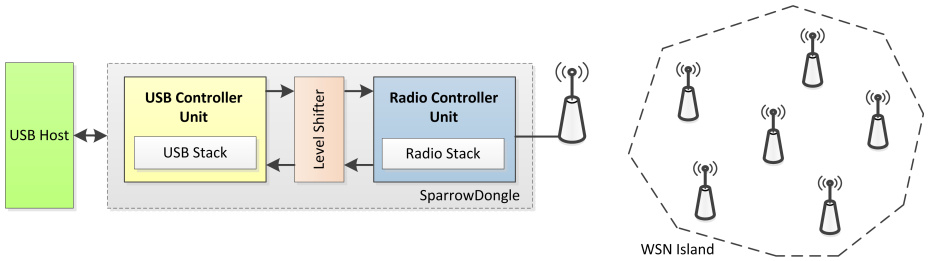

Figure 3.3: *SparrowDongle stick architecture*

with ACM module (USB Communications Device Class with Abstract Control Mode).

The dongle uses Atmega32U4 as a dedicated USB Controller unit. This design allows the RF controller to run any RF communication stack without having the USB code intrude on key timings.[VTD13]

3.2.2 The SparrowV3.2

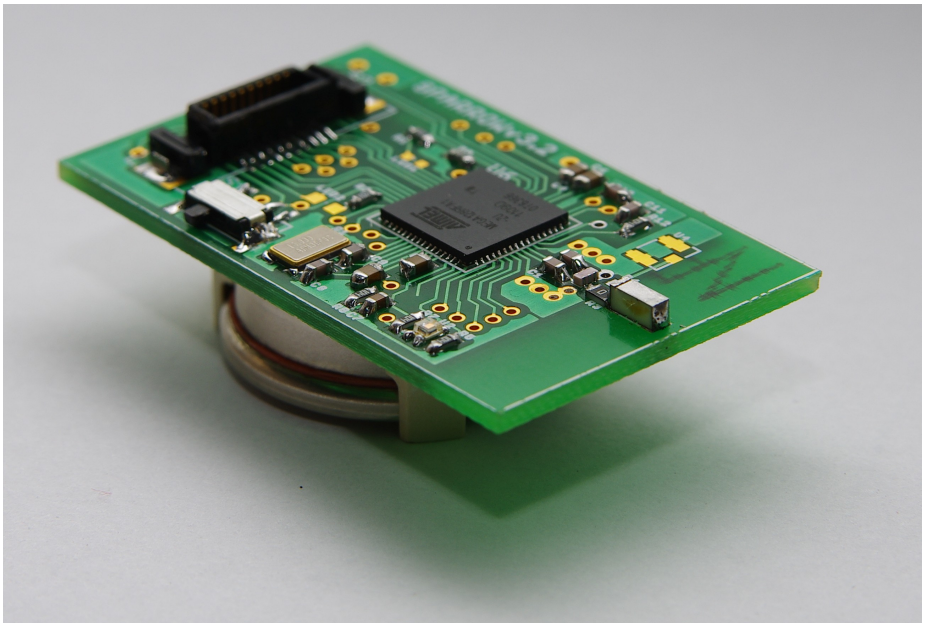

Figure 3.4: *The SparrowV3.2*

The other member of The Sparrow Family, the SparrowV32 nodes are responsible of collecting data from the surrounding environment and sending it to the Sparrow Dongle. The collected data depends on the sensors attached to the Spar-

rowV3.2. The standard implementation has a light, temperature and humidity sensor. Besides this information, it also sends the battery level.

The SparrowV32 can be modiffied by adding aditional sensors for a better understanding of the environment.

Chapter 4

Software Implementation

The solution is divided into different modules that run independently but communicate with each other to achieve the main goal. The separation of software modules allows for future features to be added easily.

The main modules are installed in the AR Parrot Drone and the Android FreeFlight 2.0 application.

The SparrowDongle gateway is always in a listen-for-data state and dumps any data received on the serial. When it receives the data, it sends back an ACK message back to let the SparrowV3.2 node to know that it can begin sending the entire stored data to the mobile gateway.

The SparrowV3.2 node is sending periodically a small data packet to check if a gateway is available. It stores the data accumulated over the period when no reply is given to it. When it receives the ACK message from a mobile gateway it starts sending the stored data to the gateway. The data sent can vary, from sensor readings to debugging information in order to check the state of the Wireless Sensor Network.

The data gathered by the gateway is saved into different files in the AR Parrot Drone's internal memory. The files also contains information such as the node identification tag and time of the transfer. The data can be accessed at any time by any device connected to the drone's wireless network port 4242 via FTP.

The drone also processes some of the collected data to provide realtime HUD information, such as signal strength, last connection time and number of discovered nodes. This information is sent to the controlling device through a socket connection.

The controlling device, pc or android, will gather the information and display them to the user.

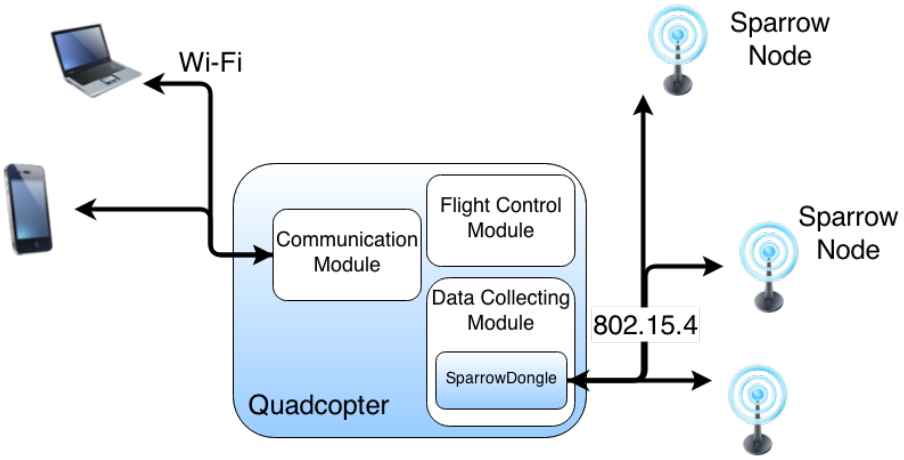

Figure 4.1: *Modules and connections between them and devices*

4.1 The Debug Module

When performing modification to the existing code, a debug enabling option whould speed up the process. The moudle whould allow for displaying controll message to the user console even when the process is running in background.If the messages would be activated all the time, they would slow down the execution speed of the process.

In order to see those messages, the debug option must be activated and then this simple command whill show them:

Listing 4.1: Simple display message command

```
p=$(pidof read) \&\& strace -p $p
```

Enabling the debug is just a matter of setting a define from 0 to 1, recompile and upload the code to the drone to see the messages.

Listing 4.2: Data Collection use of mutex

```
/* activates/deactivates printf debug information*/
#define DEBUG_ON 0
/* delay yime in microseconds*/
#define DELAY_US 100000
#define DEBUG_PRINT(a...) { if(DEBUG_ON) printf(a); }
```

Also, in certain parts of the modules, a wait action is needed in order to wait for an action to be executed. The value can be changed to any level, but you must be carefull in doing this. A small delay will send data more often but it

could use alot of processing power, while a big delay could be to slow for the data to be usable. Now the delay is set at 100 ms, for a 10 times per second data update.

4.2 The Data Collecting Module

The module saves the collected data into the drone's internal memory and pases the data on to the communication module, which will display on the controller interface certain data: number of nodes currently connected to the Dongle, de signal strength, if the Dongle is connected etc.

This module, besides the main purpose and similar to the other modules, has some extra features that are design to make the solution more user friendly and easier to improve in the feature.

4.2.1 Modules intercommunication

The memory area in which the information sent to the user is saved is shared between this module and the communication module. The interaction method between these two modules belongs to the consumer-producer archetype, where the Data Collecting Module can be associated with the producer side and the Communication Module with the consumer side.

A sensible issue with this approach regards possible deadlocks. This is prevented with the use of a mutex construct that allows only one thread at a time to modify the data.

Listing 4.3: Data Collection use of mutex

```
pthread_mutex_lock(&data_lock);
add_node_data(get_current_timestamp(),read_data + 7);
pthread_mutex_unlock(&data_lock);
```

The mutex is used similarly in the Communication Module when it consumes the information.

4.2.2 Fault tolerance

Because the Dongle is connected to an USB port on a machine that has a lot of vibrations, it might desconnect / reconnect for a very short period of time, so this module has been designed with multiple USB disconnects and reconnects without the need to reset the Drone. This infomation is vital, because you can check if the Dongle is still connected to the dron without the need to inspect it visualy or to connect to a debug terminal.

4.3 The Communication Module

All of the information gathered by the Data Collecting Module would be useless if it cannot be accessed easily.

This module, as the name suggests, handles the communication of this this crucial information back to the user.

Being a different module, with different attributions than the Data Collecting Module, it has an entire Linux process dedicated to it for 3 important reasons:

1. The approach of having a process per module allows the modules to run independently of each other;
2. The Data Collecting Module can collect the data from the Dongle as soon as this is available;
3. If the Communication Module stops working, the Data Collecting Module can keep collecting data, so complete failure of the system is avoided;
4. System processes can be restarted in case of failure.

4.3.1 Socket with connection reset

The communication is done through socket connections listening on port 8888. The server running on the drone accepts only one connection at a time.

If a connected client decides to disconnect before or while a write operation is in progress, a SIGPIPE error signal will be thrown, stopping all the modules. This is prevented by ignoring the signal, forcing the write action to return a EPIPE, and exiting gracefully.

The main process will use the callback `accept_socket_connection` to reestablish a new connection. Once a connection is established, it will send information once every DELAY_US microseconds. The program was configured and tested with a 100 ms wait period that leads to a ten times per second information update.

This delay is required because:

- If data is send too often, the socket might be flooded and stop sending the data.
- If there was no delay, it would occupy too much processor time both for the drone and the controlling device.

4.3.2 JSON Encoding of Data

[jso]

JSON is an open-standard that uses text to encode data and it is an alternative to XML.

JSON is best suitable for this application as a data encode format becuase it is data oriented, unlike XML which is document oriented.

Also it is very easy to encode, the result is smaller than the XML alternative and all devices can decode it.

The informations encoded are

- Dongle connection status
- An array containing node data
  - Node unique ID
  - Last connection time of the node to Dongle
  - The power of the received signal

4.4 Android application modules

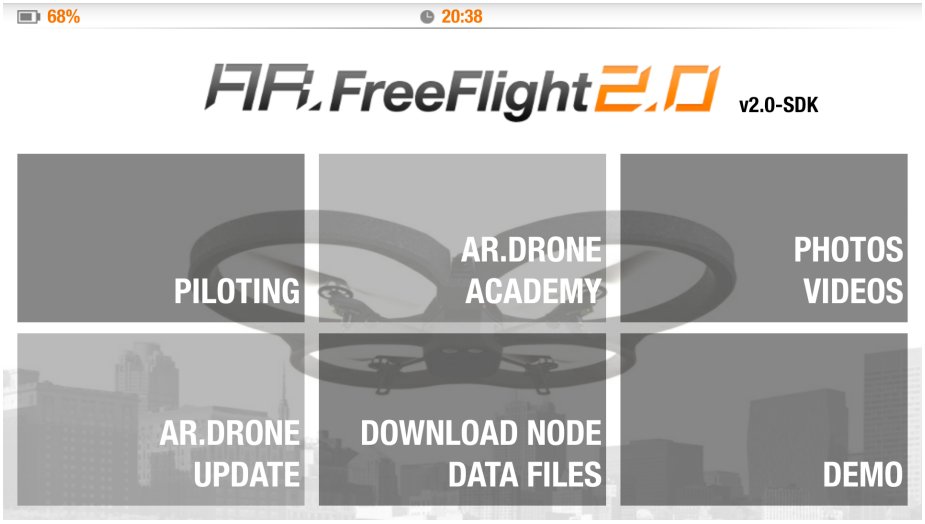

Figure 4.2: *ARFreeflight modified application*

Being an open-source platform we have modified the AR Freeflight 2.0 Android application to communication with our new modules added to the drone.

Android fairly imposes the use of the background process class AsynkTask when you have to use communication protocols like http, ftp, sockets because this prevents the UI process from being stuck in communication and not responding to user actions.

The class offers 5 very important methods that can be overwritten, 3 running on the main UI process, that prepare data before and after execution, publish the progress or simply cancel at any step, and 1 running on the actual background process.

4.4.1 Display information module

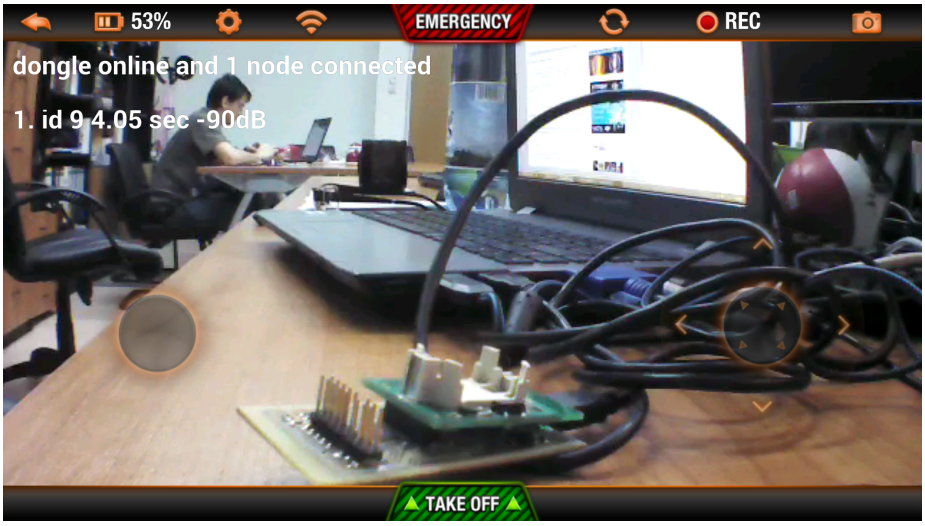

Figure 4.3: *ARFreeflight modified Piloting Screen*

The Piloting screen of the application has been modified to the display the received data from the drone.

The information displayed consists of the dongle being functional and at most, 9 nodes sorted descending after their signal strength.

4.4.2 FTP communication module

The drone has a built-in FTP server that can be configured to allow access to any folders/files on the drone. We have configured the drone so that the folder that contains the saved data can be accessed at any time using the 4242 port by any device that has FTP client capabilities. We have added this feature to the Android application as well.

The application will download all the files from the drone to the local storage of the user's Android device while displaying the progress.

W

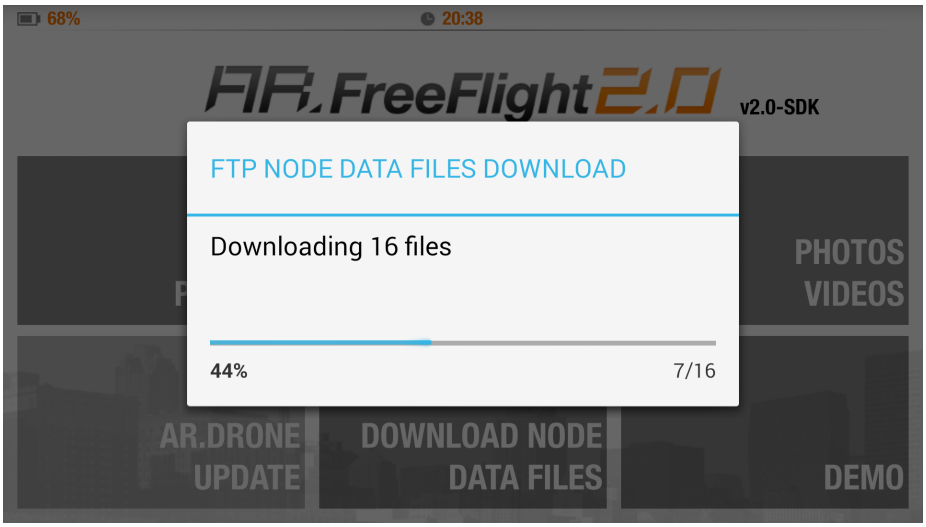

Figure 4.4: *ARFreeflight FTP downloading files*

Chapter 5

Testing

5.1 Scenario

5.2 Results

Chapter 6

Future Work

Even though the solution is working, new application and functionalities can be added. The application can be from opposing worlds, such as search and rescue versus military.

6.1 Features

6.1.1 Debug

A feature that can be used in conventional wireless sensor networks is to determine the source of communication failure. If the gateway detects that the network has a communication problem and not all of the previous nodes can be reached, this information will be send to the drone and it will try to find and determine which nodes are working properly and which nodes are not.

6.1.2 Truly Autonomous flight

The AR Parrot Drone 2.0 can perform autonomous flight with a GPS module, but only while it is still in the range of the WiFi connection.

6.2 Applications

6.2.1 Treasure Hunt Games

A treasure hunt game can be played by placing the nodes with information regarding the location of the treasure at certain hidden spots and trying to find the treasure by following the clues provided by the drone.

6.2.2 Search and Rescue

When tourists go hiking or other similar activities and get lost they can be found if they are wearing a wireless sensor that monitor vitals and can gather a periodical GPS position. The drone will search for the signal emitted by the node and help pinpointing their exact location while offering a live video feed of them.

Chapter 7

Conclusions

7.1 Outlook

Bibliography

[BRA13] ALEX BRACETTI. The 10 Best Drones You Can Buy Right Now. http://www.complex.com/pop-culture/2013/03/10-cool-drones-you-can-buy-right-now/parrot-ardrone-20, March 2013. [cited at p. 6]

[EHHW08] Aysegül Tüysüz Erman, LV Hoesel, Paul Havinga, and Jian Wu. Enabling mobility in heterogeneous wireless sensor networks cooperating with uavs for mission-critical management. *Wireless Communications, IEEE*, 15(6):38–46, 2008. [cited at p. 5]

[jso] Json Standard. http://www.json.org/xml.html. [cited at p. 14]

[OBLC+07] Anibal Ollero, Markus Bernard, Marco La Civita, Lodewijk van Hoesel, Pedro J Marron, Jason Lepley, and Eduardo de Andres. Aware: Platform for autonomous self-deploying and operation of wireless sensor-actuator networks cooperating with unmanned aerial vehicles. In *Safety, Security and Rescue Robotics, 2007. SSRR 2007. IEEE International Workshop on*, pages 1–6. IEEE, 2007. [cited at p. 5]

[Par12] AR Parrot. Drone. *Available: ardrone. parrot. com*, 75, 2012. [cited at p. 6]

[VSB+11] João Valente, David Sanz, Antonio Barrientos, Jaime del Cerro, Ángela Ribeiro, and Claudio Rossi. An air-ground wireless sensor network for crop monitoring. *Sensors*, 11(6):6088–6108, 2011. [cited at p. 4]

[VTD13] Andrei Voinescu, Dan Tudose, and Dan Dragomir. A lightweight, versatile gateway platform for wireless sensor networks. In *Networking in Education and Research, 2013 RoEduNet International Conference 12th Edition*, pages 1–4. IEEE, 2013. [cited at p. 9]

List of Figures

List of Tables

Listings